⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平2−151930

⑤1Int. Cl.⁵	識別記号	庁内整理番号
G 06 F 9/38	350 X	7361−5B
12/00	303 G	8841−5B

⑬公開　平成2年(1990)6月11日

審査請求　未請求　請求項の数 2　(全11頁)

⑤4発明の名称　ストアバッファ管理方式

⑳特　願　昭63−306137
㉒出　願　昭63(1988)12月5日

⑦2発明者　宮沢　文彦　東京都港区芝5丁目33番1号　日本電気株式会社内
⑦1出願人　日本電気株式会社　東京都港区芝5丁目33番1号
⑦4代理人　弁理士　山川　政樹　外2名

明　細　書

1. 発明の名称

ストアバッファ管理方式

2. 特許請求の範囲

(1) ストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算装置と主記憶装置とを含む情報処理装置において、前記ストアバッファに登録中であり前記主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のストアを該ロード命令の処理に優先して前記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、該ストア処理完了まで前記ロード命令の待合わせを行う手段と、前記ロード命令と同一主記憶アドレスへのストア処理が終了したならば前記ロード命令の待合わせの解除を行う手段とを含むことを特徴とするストアバッファ管理方式。

(2) ストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算処理と主記憶装置とを含む情報処理装置において、ストアバッファに登録中であり主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行の該ストア命令のみを前記ロード命令の処理に優先して前記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、該ストア処理完了まで前記ロード命令の待合わせを行う手段と、前記ロード命令と同一主記憶アドレスへのストア処理を終了したならば前記ロード命令抑止の解除を行う手段と、すでに主記憶装置に書込んでしまつたストアバッファ内実行済みの該ストア命令に対しては再度主記憶装置に対して書込むことがないように制御する手段とを含むことを特徴とするストアバッファ管理方式。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明はストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算装置と主記憶装置とを含む情報処理装置における命令の

高速化処理に係り、特にストア命令の高速処理を実現するためのストアバッファ管理方式に関するものである。

〔従来の技術〕

従来のパイプライン型情報処理装置においては、ロード命令処理が優先的に主記憶との間で処理を行ないストア命令処理はストアバッファに格納された後で、ロード命令が主記憶とアクセスを行なつていない間に主記憶に対しての書込み処理を行なう。そして、先行するストア命令と同一主記憶アドレスに対して後続ロード命令が存在した場合、ストア命令処理をロード命令処理に優先して処理しなければ、主記憶のデータに誤まりが生じてしまう。

そこで、主記憶の同一アドレスへの先行ストア命令に対し後続ロード命令が存在する場合には、ストアバッファ内の未実行のストア命令処理をそのロード命令処理よりも優先して処理を行ない、ストアバッファ内の全ての未実行のストア処理が終了したならば、そのロード命令処理を実行するようになつていた。

〔発明が解決しようとする課題〕

上述した従来のストアバッファ管理方式では、ストアバッファに登録中であり未実行であるストアドレスに対し後続するロード要求が同一アドレスに対し実行しようとした場合、ストアバッファ内に登録されている個々のストアアドレスに対し、どのストアアドレスがそのロード要求と一致しているのかを示す手段を持つていなかつたために先行ストアアドレスと後続ロードアドレスが一致した場合にはストアバッファ内のすべての主記憶に対して未実行のストア処理をそのロード要求に優先して処理しなければならず、そのロード要求の処理を開始する迄に時間を費やすという課題があつた。

〔課題を解決するための手段〕

本発明のストアバッファ管理方式は、ストアバッファを有し、ストア処理を必要とする命令の実行をサポートする演算装置と主記憶装置とを含む情報処理装置において、上記ストアバッファに登録中であり、上記主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のストアをそのロード命令の処理に優先して上記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、そのストア処理完了まで上記ロード命令の待合わせを行う手段と、上記ロード命令と同一主記憶アドレスへのストア処理が終了したならば上記ロード命令の待合わせの解除を行う手段とを含むものである。

また、本発明の別の発明によるストアバッファ管理方式は、上記の情報処理装置において、ストアバッファに登録中であり主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のそのストア命令のみを上記ロード命令の処理に優先して上記ロード命令のアクセスアドレスと同一主記憶アドレスに対して実行し、そのストア処理完了まで上記ロード命令の待合わせを行う手段と、上記ロード命令と同一主記憶アドレスへのストア処理を終了したならば上記ロード命令抑止の解除を行う手段と、すでに主記憶装置に書込んでしまつたストアバッファ内実行済みのそのストア命令に対しては再度主記憶装置に対して書込むことがないように制御する手段とを含むものである。

〔作用〕

本発明においては、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、ストアバッファ内の同一主記憶アドレスに対してのストア処理が全て終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換える。また、別の発明においては、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、ストアバッファ内の同一主記憶アドレスに対してのストア処理のみを実行し処理が終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換える。

〔実施例〕

以下、図面に基づき本発明の実施例を詳細に説明する。

第１図は本発明の一実施例を示すブロック図である。

図において、１はストアバッファアドレス書込みレジスタ、２，３，４はストアアドレス登録レジスタ、５，６，７は一致検出コンパレーター、８は主記憶書込み待ちアドレスレジスタ、９は掃出し制御部、１０はストアバッファ読出しレジスタ、１１はインクリメンタ、１２はストアバッファデータ書込みレジスタ、１３はインクリメンタ、１４はストアデータバッファ、１５は主記憶書込み待ちデータレジスタ、１６はインクリメンタである。

そして、ストアバッファに登録中であり、主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のストアをそのロード命令の処理に優先してロード命

令のアクセスアドレスと同一主記憶アドレスに対して実行するように構成されている。また、一致検出コンパレータ５～７と掃出し制御部９はストア処理完了までロード命令の待合わせを行う手段を構成し、掃出し制御部９はロード命令と同一主記憶アドレスへのストア処理が終了したならばロード命令の待合わせの解除を行う手段を構成している。

第２図は第１図における掃出し制御部９の詳細を示す図である。

この第２図において、１７はフラグバッファレジスタ、１８はアダー、１９はレジスタ、２０はデクリメンタである。

第３図は通常ストア処理のタイムチャートで、(f)は「第一例」を示し、(n)は「第二例」を示す。

そして、(a)はサイクルを示したものであり、(b)はステージＩＦ、(c)はステージＡＣ、(d)はステージＡＴ、(e)はステージＣＡ、(f)はステージＥＸ、(g)はステージＳＴ、(h)はストアアドレス登録レジスタＳＡ、(i)は主記憶を示したものである。

第４図は本発明の一実施例のタイムチャートで、(a)はサイクルを示したものであり、(b)はステージＩＦ、(c)はステージＡＣ、(d)はステージＡＴ、(e)はステージＣＡ、(f)はステージＥＸ、(g)はステージＳＴ、(h)はストアアドレス登録レジスタＳＡ、(i)は主記憶、(j)はライトアドレスポインタ、(k)はライトデータポインタ、(l)はリードポインタを示したものである。

第５図はパイプライン処理の一例を示す図である。

つぎに第１図に示す実施例の動作を第２図ないし第５図を参照して説明する。

パイプライン処理の一例を示す第５図においては、ステージＩＦからステージＳＴまで６ステージに分割されている。そして、ステージＩＦは命令を取出す処理、ステージＡＣはオペランドを生成する処理、ステージＡＴは論理アドレスを実アドレスに変換する処理、ステージＣＡはオペランドを読出す処理、ステージＥＸは読出されたオペラドの演算処理、ステージＳＴは演算結果の格納処

理を行う。

以降本発明の実施例をこの第５図に示すパイプラインステージを基にして説明する。

まず、通常ストア処理について第３図に示すタイムチャートを参照して説明する。(f)に示す「第一例」として時間 t_1 においてストアＡ命令処理が開始されると、時間 t_4 においてストアＡに関するストアアドレスが第１図のストアアドレス登録レジスタ２～４に登録される。そして、時間 t_5 において演算が行なわれ、時間 t_6 においてストアすべきデータがステージＳＴの第１図に示すストアデータバッファ１４に登録される。このように、ストア命令においてはアドレスがストアバッファに登録されるタイミングとストアデータがストアバッファに登録するタイミングが異なつている。そして、時間 t_6 においてアドレスＢに対してのロードＢ命令がステージＡＴに存在したとすると、本発明における情報処理装置においてはロード命令処理がストア命令処理よりも優先して処理されるように制御されているために、時間 t_7 に

おいてロードB命令が主記憶とのアクセスを行ない、ストアA命令はストアバッファで処理の待合わせを行ない、時間t_6においてストアA命令の主記憶とのアクセスが行なわれる。

そして、(b)に示す「第二例」として時間t_7においてストアC命令の処理が始まり、時間t_{10}においてロードC命令の処理が始まるとすると、時間t_{12}において先行するストアC命令が完了しないうちに後続ロードC命令を実行しようとする。ここで、後続ロードC命令が先に処理されてしまうならば主記憶から間違ったデータを取出してしまう。そこで、先行するストアC命令を優先する必要があり、時間t_{13}においてストアC命令の主記憶とのアクセスが行なわれ、時間t_{14}においてロードC命令の主記憶とのアクセスが行なわれる。

本発明は第3図の(b)に示す「第二例」における制御方法であり、第1図に示す実施例および第4図に示すタイムチャートにより具体的に説明する。

第4図のタイムチャートにおいて、ストアD，

ストアE，ストアE，ストアF，ロードG，ロードH，ロードEの順に時間t_1～t_7において各命令がステージIFに受付けられるとすると、ストア命令は、

IF → AC → AT → CA → EX → ST

と処理が進み、ロード命令は

IF → AC → AT

と処理が進む。そして、ストア命令においては、AT → CA のタイミングで各ストア命令のアドレスがステージSAに登録される（t_4～t_7）。EX → ST のタイミングでストア命令のストアデータが第1図のストアデータバッファ14に登録される（t_5～t_8）。そして、ストアアドレスとストアデータが揃つたストア命令はストアバッファ→主記憶への掃出しタイミングを待つ。この掃出しタイミングは「ロード命令が主記憶を使用しない時」である。よつて、ロード命令がステージAT → 主記憶へ掃出すタイミング以外のときにストアバッファから主記憶への掃出しが可能となる。

そして、時間t_7ではストアDが主記憶に掃出

され、時間t_8ではロードGの主記憶アクセスが行なわれ、時間t_9ではロードHの主記憶アクセスが行なわれる。

時間t_8のステージATにアドレスEに対するロードE命令が存在する。また、ストアバッファ内にはアドレスEに対するストアE命令が存在する。このとき、「第二例」で説明したように、ストアバッファ内から未実行のストアE命令が終了するまでロードE命令の処理よりもストア命令処理を優先させる。よつて、時間t_{10}においてストアEが主記憶に掃出され、時間t_{11}において次のストアEが主記憶に掃出される。この時点でストアバッファ内には未実行のストアE命令はなくなるので、第1図に示す掃出し制御部9においてストアバッファの掃出し抑止を行い、時間t_{12}においてロードE命令の主記憶アクセスが行なわれ、時間t_{13}においてストアFの主記憶掃出しが行なわれる。

そして、ストア命令の第1図に示すストアアドレス登録レジスタ2～4への書込みはストアバッ

ファアドレス書込みレジスタ1によつて，ストアデータバッファ14への書込みはストアバッファデータ書込みレジスタ12によつてそれぞれ行なわれ、ストアバッファから主記憶への掃出しはストアバッファ読出しレジスタ10によつて行なわれる。

掃出し制御部9においては、ストアバッファ内に未実行の、「ロード命令よりも優先しなければならないストア命令」がいくつ存在するかが把握されている。この掃出し制御部9の詳細を第2図に示す。第1図において各ストアアドレス登録レジスタ2～4の値と後続アドレスの値が一致検出コンパレータ5～7によつて一致が調べられ、もし一致しており、後続命令がロード命令であるならば第2図のフラグバッファレジスタ17にフラグがセットされる。また、一致検出コンパレータ5～7のTotal数、つまり、掃出さなければならないストア命令数がレジスタ19に示される。そして、第1図のストアバッファ読出しレジスタ10によつてストアバッファの掃出しが行なわれ、

第２図のフラグバッファレジスタ１７からの出力が「1」であるならば－１カウンタであるデクリメンタ２０によりレジスタ１９の減算を行い、このレジスタ１９が「0」になつたならばストアバッファ内に掃出し必要なストア命令は無いとしてストアバッファからの掃出しを抑止し，ロード命令を開始するように制御の変更を行なう。

第６図は本発明の他の実施例を示すブロック図である。

この第６図において第１図と同一符号のものは相当部分を示し、２１は掃出し制御部である。

そして、ストアバッファに登録中であり、主記憶装置に対して未実行であるストア命令のアドレスと一致する後続ロード命令のアクセスがあつた際、ストアバッファに登録されている未実行のそのストア命令のみをロード命令の処理に優先してロード命令のアクセスアドレスと同一主記憶アドレスに対して実行するように構成されている。また、一致検出コンパレーター５～７と掃出し制御部２１はストア処理完了までロード命令の待合わ

せを行う手段を構成し、掃出し制御部２１はロード命令と同一主記憶アドレスへのストア処理を終了したならばロード命令抑止の解除を行う手段を構成すると共に、すでに主記憶装置に書込んでしまつたストアバッファ内実行済みのそのストア命令に対しては再度主記憶装置に対して書込むことがないように制御する手段を構成している。

第７図は第６図における掃出し制御部２１の詳細を示す図である。

この第７図において２２はデコーダ、２３は有効ストア指示フラグバッファ、２４はエンコーダ、２５は優先ストア指示フラグバッファ、２６はエンコーダ、２７はストアバッファ読出し指定レジスタ２８はセレクタ、２９はストアバッファ読出しレジスタ、３０はアダー、３１はレジスタ、３２はデクリメンタ、33_1，33_2 …33_n は論理積回路、３４は優先ストア指示信号線である。

第８図は本発明の他の実施例のタイムチャートで、(a)はサイクルを示したものであり、(b)はステージＩＦ、(c)はステージＡＣ、(d)はステージＡＴ、(e)はステージ ＣＡ、

(f)はステージＥＸ、(g)はステージＳＴ、(h)はストアアドレス登録レジスタＳＡ、(i)は主記憶、(j)はライトアドレスポインタ、(k)はライトデータポインタ、(l)はリードポインタを示したものである。

つぎに第６図に示す実施例の動作を第７図および第８図を参照して説明する。なお、パイプライン処理については前述の第１図に示す実施例と変わらないので、ここでの説明を省略する。

まず、本発明は第３図の(ロ)に示す「第二例」における制御方法である。

第８図に示すタイムチャートにおいて、ストアＤ，ストアＥ，ストアＦ，ストアＥ，ロードＧ，ロードＨ，ロードＥ，ストアＧの順に時間t_1からt_9において各命令がステージＩＦに受付けられるとすると、ストア命令は

ＩＦ→ＡＣ→ＡＴ→ＣＡ→ＥＸ→ＳＴ

と処理が進み、ロード命令は

ＩＦ→ＡＣ→ＡＴ

と処理が進む。そして、ストア命令においては

ＡＴ→ＣＡのタイミングで各ストア命令のアドレ

スが第６図のストアアドレス登録レジスタ２～４に登録される（t_4～t_7，t_{11}）。また、ＥＸ→ＳＴのタイミングでストア命令のストアデータが第６図のストアデータバッファ１４に登録される（t_5～t_8，t_{12}）。

そして、ストアアドレスとストアデータが揃つたストア命令はストアバッファから主記憶への掃き出しタイミングを待つ。この掃き出しタイミングは「ロード命令が主記憶を使用しない時」である。よつて、ロード命令がステージＡＴから主記憶へアクセスするタイミング以外のときにストアバッファから主記憶への掃き出しが可能となる。

時間t_7ではストアＤが主記憶に掃き出され、時間t_8ではロードＧの主記憶アクセスが行なわれ、時間t_9ではロードＨの主記憶アクセスが行なわれる。時間t_9のステージＡＴに主記憶アドレスＥに対するロードＥ命令が存在する。このとき、ストアバッファ内には主記憶アドレスＥに対するストアＥ命令が存在する。このとき第３図の(ロ)に示す「第二例」で述べたように、ストアバッ

フア内から主記憶に対して未実行のストアＥ命令が終了するまでロードＥ命令の処理よりもストアＥ命令処理を優先させる。

このとき、この第６図に示す実施例においては、ストアバッファ内に主記憶アドレスＥに対するストア命令は２つ存在するため、この２つのストア命令Ｅの実行を優先的に行なう。そして、時間 t_{10} において最初のストアＥ命令がストアバッファから主記憶に掃き出され、時間 t_{11} において２番目のストアＥ命令が主記憶に掃き出される。この時点でストアバッファ内には主記憶に対して未実行のストアＥ命令はなくなるので、掃出し制御部２１においてストアバッファの掃き出し抑止を行ない、時間 t_{12} においてロードＥ命令の主記憶アクセスが行なわれ、時間 t_{13} においてストアバッファ内の主記憶に対して未実行のストアＦ命令が行なわれ、時間 t_{14} においてはストアＧ命令が主記憶アドレスＧに対して実行される。

そして、ストア命令のストアアドレス登録レジスタ２～４への書込みは、ストアバッファアドレス書込みレジスタ１によつて，ストアデータバッファ１４への書込みはストアバッファデータ書込みレジスタ１２によつてそれぞれ行なわれ、ストアバッファから主記憶への掃き出しは第７図のストアバッファ読み出しレジスタ２９によつて行なわれる。

掃出し制御部２１においては、ストアバッファ内に未実行の「ロード命令よりも優先しなければならないストア命令」がいくつ存在するか把握しており、各ワードに対し掃き出さなくてはならない命令であるかをフラグを立てることによつて示す。このフラグが立つているストア命令についてのみストアバッファからの掃き出しを行うように掃出し制御部２１で制御する。また、そのストア処理終了後、ストアバッファ内の主記憶に対して未実行であるストア命令のワード位置にストアバッファ読出しレジスタの値を戻しておく必要があり、すでに主記憶に対して掃き出してしまつたストア命令に対しては再度ストアすることがないように掃出し制御部２１で制御を行う。

第７図にこの掃出し制御部２１の実施例を示す。

通常ストア処理時は有効ストア指示フラグバッファ２３を参照する。そして、この有効ストア指示フラグバッファ２３はストアアドレス書込み時に指定ワードにフラグ「1」を立て、ストアバッファ掃き出し時にリセットを行なうフラグバッファで、各ワードのストア命令が主記憶に対して実行済であるか，未実行であるかを示す。

また、ストアバッファ読出し指定レジスタバッファ２７には０～Ｎまでストアバッファのワード数の値がそれぞれにセットされている。そして、有効ストア指示フラグバッファ２３からの出力信号をエンコーダ２４によつてストアバッファ内未実行のストア命令のワード位置を指定するようにエンコードを行ないセレクタ２８によりストアバッファ読出し指定レジスタバッファ２７からの出力信号のうちストアバッファから掃き出すべきワード位置を指定してストアバッファ読出しレジスタ２９のセットを行う。

そして、前述の第３図の(b)の「第二例」に示すような同一記憶アドレスに対して未実行のストア命令とロード命令が競合する場合には、論理積回路 33_1～33_n において第６図に示す一致検出コンパレーター５～７の値が「1」であり（先行ストアアドレス＝後続ロードアドレス）かつロードリクエスト処理要求時であり，さらに主記憶に対して未実行のストアリクエスト（有効ストア指示フラグバッファ２３の出力）である場合、優先ストア指示フラグバッファ２５の対応ワードにフラグを立てる。この優先ストア指示フラグバッファ２５の出力信号をエンコーダ２６によつてデコードを行ない、このエンコーダ２６によりどのストア命令処理を行なうかの判断を行ないセレクタ２８を通つてストアバッファ読出し指定レジスタバッファ２７からの出力信号を選択しストアバッファ読出しレジスタ２９のセットを行う。そして、アダー３０においては優先処理すべきストア命令がストアバッファ内にいくつ存在するかを計算しており、レジスタ３１にセットする。また、優先ストア処理が１つ実行される度にデクリメンタ３２に

よつてレジスタ３１の更新を行なう。そして、レジスタ３１が「0」である場合は通常ストア処理時であり、レジスタ３１が「1」である場合には同一主記憶アドレスへのアクセスに関しストアバッファ内未実行ストア命令とロード命令が競合する場合であり、セレクタ２８の選択信号に使われる。

有効ストア指示フラグバッファ２３のリセットはストアバッファ読出しレジスタ２９にセットされようとするワード位置のフラグに対し行なわれる。そして、優先ストア指示フラグバッファ２５から一度読出されたワード位置のフラグは同一タイミングでリセットが行なわれる。つまり、有効ストア指示フラグバッファ２３内および優先ストア指示フラグバッファ２５内には主記憶に対して未実行のストア命令のワード位置のフラグのみが「1」となつている。

〔発明の効果〕

以上説明したように本発明は、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、

ストアバッファ内の同一主記憶アドレスに対してのストア処理が全て終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換えることにより、不必要なストア処理によりロード命令の主記憶とのアクセスが遅れることがなくなるという効果がある。

また、本発明は、同一主記憶アドレスに対して先行する未実行のストアバッファ内ストア命令に対し後続ロード命令が存在する場合、ストアバッファ内の同一主記憶アドレスに対してのストア処理のみを実行し処理が終了した時点でストア処理をロード処理に優先していたものをロード処理優先に切換えることにより、不必要なストア処理によりロード命令の主記憶とのアクセスが遅れることがなくなるという効果がある。

4. 図面の簡単な説明

第１図は本発明の一実施例を示すブロック図、第２図は第１図における掃出し制御部の詳細を示す図、第３図は通常ストア処理のタイムチャート、第４図は本発明の実施例のタイムチャート、第５

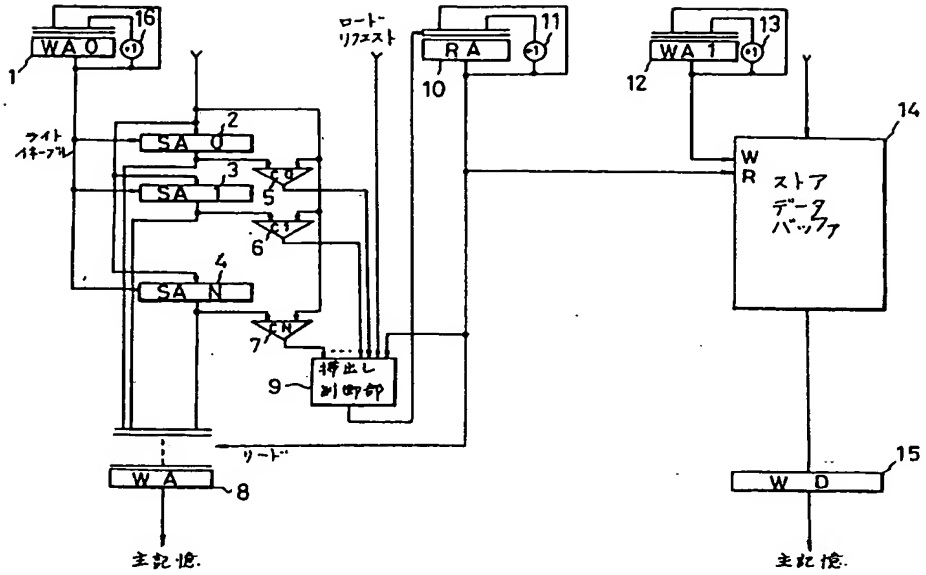
図はパイプライン処理の一例を示す図、第６図は本発明の他の実施例を示すブロック図、第７図は第６図における掃出し制御部の詳細を示す図、第８図は本発明の他の実施例のタイムチャートである。

１・・・・ストアバッファアドレス書込みレジスタ、２～４・・・・ストアアドレス登録レジスタ、５～７・・・・一致検出コンパレーター、８・・・・主記憶書込み待ちアドレスレジスタ、９・・・・掃出し制御部、１０・・・・ストアバッファ読出しレジスタ、１１・・・・インクリメンタ、１２・・・・ストアバッファデータ書込みレジスタ、１３・・・・インクリメンタ、１４・・・・ストアデータバッファ、１５・・・・主記憶書込み待ちデータレジスタ、１６・・・・インクリメンタ、１７・・・・フラグバッファレジスタ、１８・・・・アダー、１９・・・・レジスタ、２０・・・・デクリメンタ、２１・・・・掃出し制御部、２２・・・・デコーダ、２３・・・・有効ストア指示フラグバッファ、２４・・・・エンコーダ、２５・・・・優先ストア指示フラグバッファ、２６・・・・エンコーダ、２７・・・・ストアバッファ読出し指定レジスタ、２８・・・・セレクタ、２９・・・・ストアバッファ読出しレジスタ、３０・・・・アダー、３１・・・・レジスタ、３２・・・・デクリメンタ、33_1～33_n・・・・論理積回路。

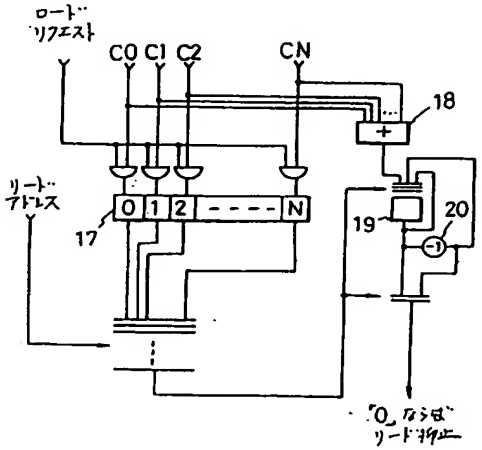
特許出願人　日本電気株式会社

代理人　山川政樹（ほか２名）

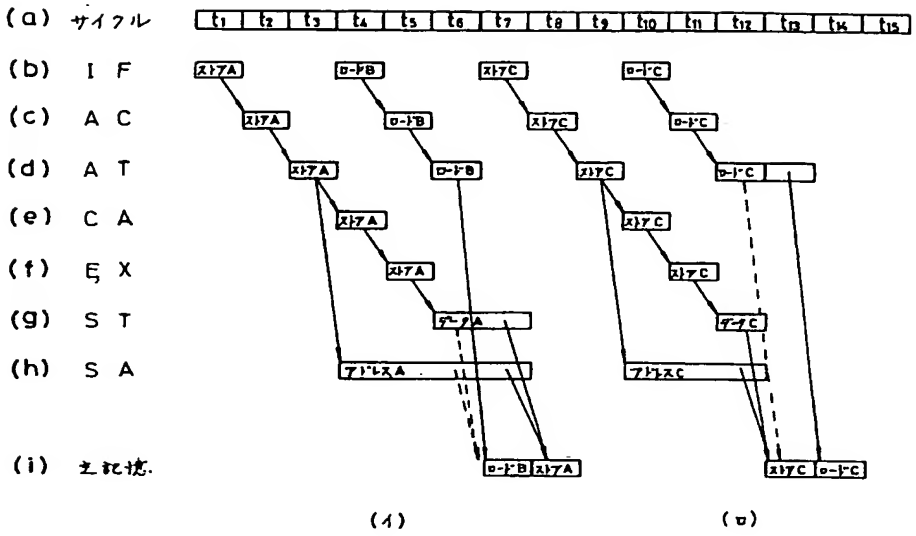
第 1 図


WA 0
RA
WA 1
SA 0
SA 1
SA N
ライトイネーブル
ロードリクエスト
ストアデータバッファ
W R
W A
W D
排出し制御部
リード
主記憶
主記憶

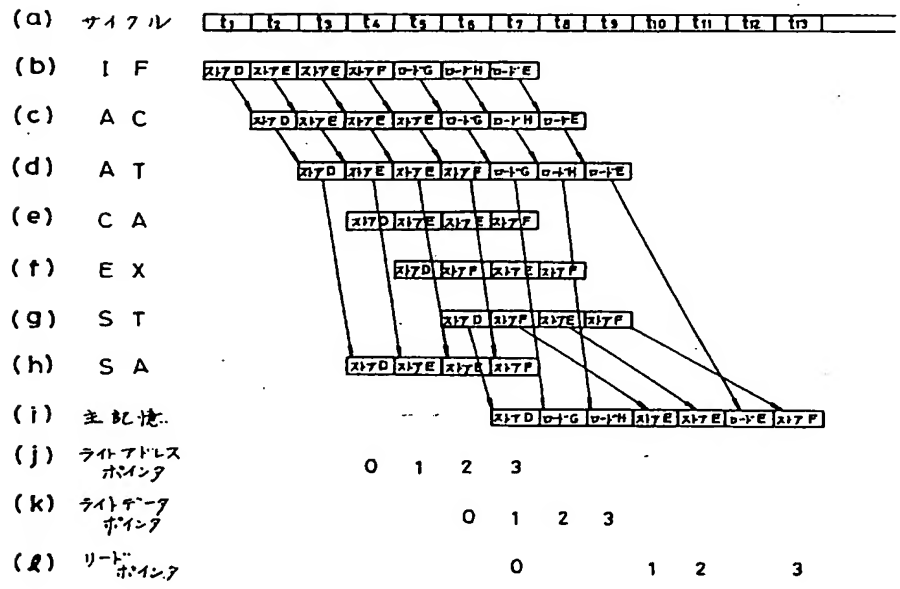
第 2 図


ロードリクエスト
C0 C1 C2 CN
リードアドレス
0 1 2 - - - - N
"0" ならばリード停止

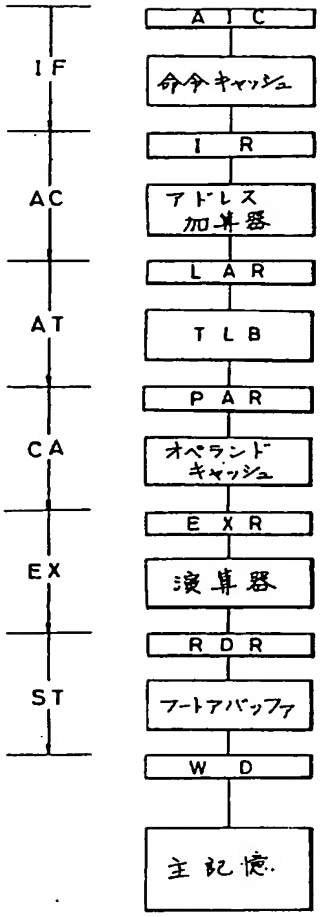
第 3 図


(a) サイクル
(b) I F
(c) A C
(d) A T
(e) C A
(f) E X
(g) S T
(h) S A
(i) 主記憶
(イ)
(ロ)

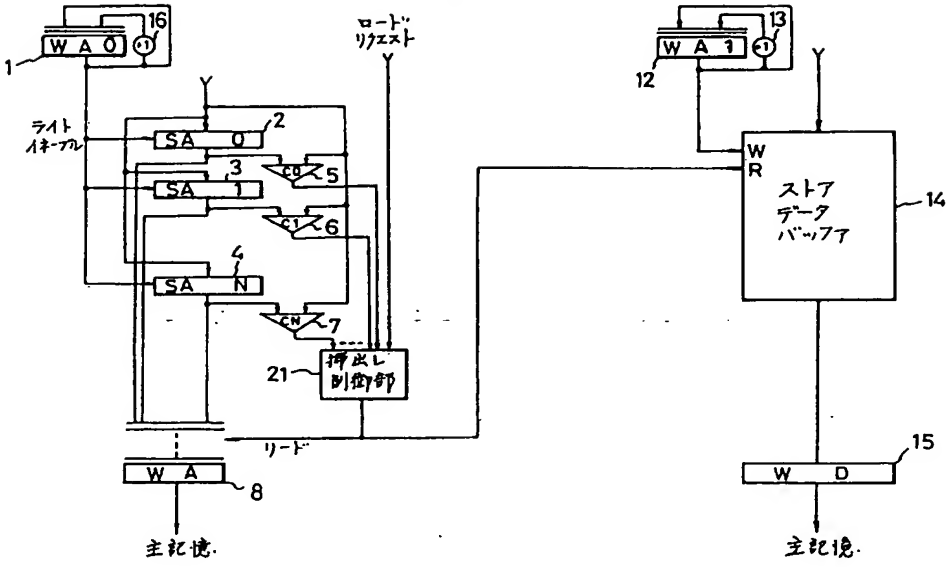
第 4 図


(a) サイクル
(b) I F
(c) A C
(d) A T
(e) C A
(f) E X
(g) S T
(h) S A
(i) 主記憶
(j) ライトアドレスポインタ
(k) ライトデータポインタ
(l) リードポインタ
0 1 2 3
0 1 2 3
0 1 2 3

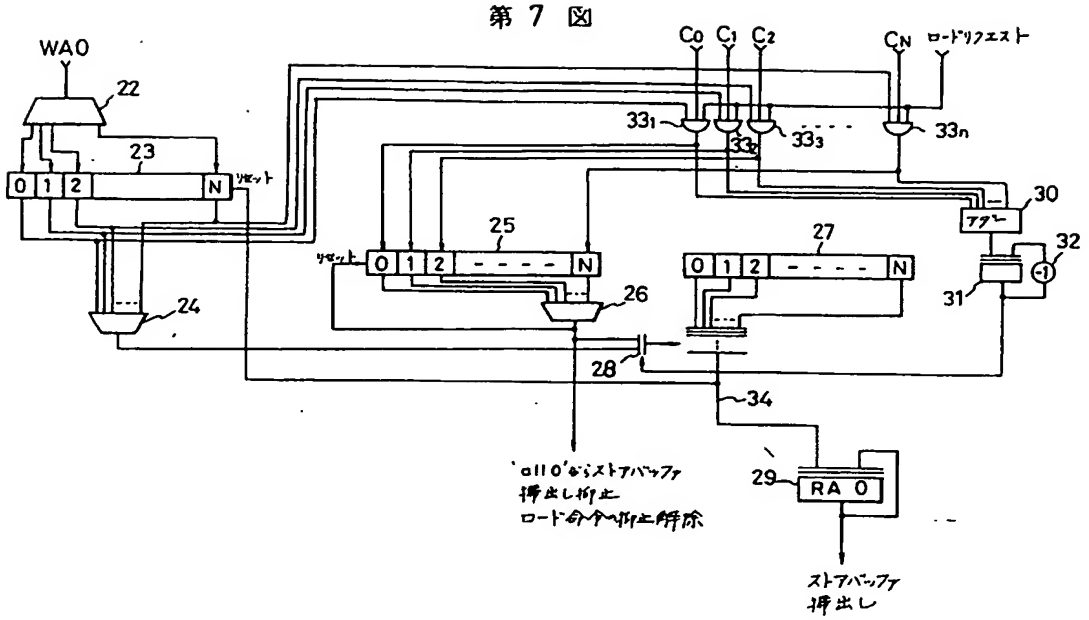
第 5 図


IF
AC
AT
CA
EX
ST
A I C
命令キャッシュ
I R
アドレス加算器
L A R
T L B
P A R
オペランドキャッシュ
E X R
演算器
R D R
フートアバッファ
W D
主記憶

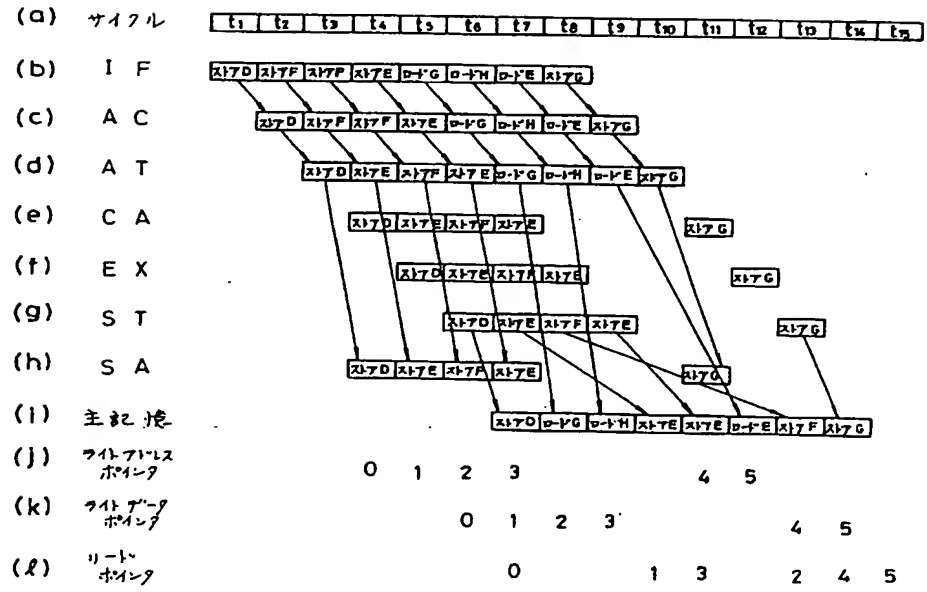
第 6 図


WA 0
16
1
ロードリクエスト
ライトイネーブル
SA 0
2
SA 1
3
SA N
4
C0
5
C1
6
CN
7
Y
押出し制御部
21
リード
WA
8
主記憶
WA 1
13
12
WR
ストアデータバッファ
14
WD
15
主記憶

第 7 図


WA0
22
23
0 1 2 N
リセット
24
C0 C1 C2
CN
ロードリクエスト
331
332
333
33n
30
アダー
32
-1
31
25
27
26
28
34
29
RA 0
all 0'sでストアバッファ
押出し抑止
ロード命令抑止解除
ストアバッファ
押出し

第 8 図


(a) サイクル
(b) I F
(c) A C
(d) A T
(e) C A
(f) E X
(g) S T
(h) S A
(i) 主記憶
(j) ライトアドレスポインタ
(k) ライトデータポインタ
(l) リードポインタ

JAPANESE LAID-OPEN PATENT APPLICATION

H02-151930 (1990)

(19) Japan Patent Office (JP) (11) Publication No. H02-151930

(12) Published Unexamined Patent Application (A) (43) Publication Date June 11, 1990

(51) Int. Cl.[5]	Ident. Code	In-House Ref. No.
G 06 F 9/38	350 X	7361-5B
12/00	303 G	8841-5B

No examination request Number of Claims 2 (totally 11 pages)

(54) Title of the Invention

STORAGE BUFFER MANAGEMENT SYSTEM

(21) Application No. PA S63-306137

(22) Date of Filing December 5, 1988 (Showa 63)

(72) Inventor Fumihiko Miyazawa
C/O Nihon Electric Corporation
5-33-1 Shiba
Minato-ku, Tokyo

(71) Applicant Nihon Electric Corporation
5-33-1 Shiba
Minato-ku, Tokyo

(74) Agent Masaki Yamakawa, Attorney
(two others)

Specification

1. Title of the Invention

Storage buffer Management System

2. Claims

Claim 1

A storage buffer management system for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, and which comprises:

a means for queuing a load instruction until the storage management is completed and executes the unexecuted storage registered in a storage buffer in relation to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer, and

a means for removing the queue of the load instruction if the storage processing is completed to the same main memory address as the load instruction.

Claim 2

A storage buffer management system for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, and which comprises:

a means for queuing a load instruction until the storage management is completed and executes only the unexecuted storage instruction registered in a storage buffer in relation

to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer; and

a means for removing the load instruction inhibitor if the storage processing is completed to the same main memory address as the load instruction; and

a means for controlling so as to prevent an executed storage instruction in the storage buffer already written to the main memory device from being written again to the main memory device.

3. Detailed Description of the Invention

Industrial Applications

The present invention for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management relates to the speeding up of processing instructions, and especially relates to a storage buffer management method for realizing high speed processing of storage instructions.

Prior Art

Regarding conventional pipeline type information processing devices, writing processing to the main memory is performed while the load instruction has not yet accessed the main memory after the load instruction processing which has priority is processed between the main memory, and the storage instruction processing is stored in the storage buffer. Further, when there is a successive load instruction in relation to the same main memory address as a preceding storage instruction, an error can occur in the data of the main

memory if the storage instruction processing is not processed having priority over the load instruction processing.

Therefore, when there is a successive load instruction in relation to a preceding storage instruction of the same address of the main memory, the unexecuted storage instruction processing within the storage buffer is processed having priority over such load instruction processing, and is constructed such that such load instruction processing is executed after all unexecuted storage processing within the storage buffer has been completed.

Problems Overcome by the Invention

With the conventional storage buffer management system described above, when a successive load request is attempted to be executed to the same address in relation to an unexecuted storage address during registration to the storage buffer, since there is no means to indicate which storage address matches such a load address in relation to each individual storage address registered in the storage buffer, there is the problem that when a preceding storage address matches a successive load address, unexecuted storage processing in relation to all of the main memory within the storage buffer must be processed as having priority over the load request; otherwise the time until initiating the processing for such load request take too long.

Problem Resolution Means

The storage buffer management system of the present invention for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, and comprises a means for queuing a load instruction until the storage management is completed and executes the unexecuted storage registered in a storage buffer in relation to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a

successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer, and

a means for removing the queue of the load instruction if the storage processing is completed to the same main memory address as the load instruction.

Further, the storage buffer management system according to a separate invention of the present invention for an information management device that has a storage buffer and which includes a main memory device and a calculating device that supports the execution of instructions necessary for storage management, comprises:

a means for queuing a load instruction until the storage management is completed and executes only the unexecuted storage instruction registered in a storage buffer in relation to the same main memory address as the access address of the load instruction that has priority in processing such load instruction when accessing a successive load instruction that matches the address of an unexecuted storage instruction in the main memory device during registration to the storage buffer, and

a means for removing the load instruction inhibitor if the storage processing is completed to the same main memory address as the load instruction, and

a means for controlling so as to prevent executed storage instructions in the storage buffer already written to the main memory device from being written again to the main memory device.

Operation

According to the present invention, when there is a successive load instruction in relation to the storage instruction within an unexecuted storage buffer that precedes the same main memory address, storage processing which has priority over the load processing is switched to load processing priority at the time of completion of all the storage processing in relation to the same memory address within the storage buffer. Further, in

another invention, when there is a successive load instructions in relation to the storage instruction within an unexecuted storage buffer that precedes the same main memory address, storage processing is only executed in relation to the same main memory address within the storage buffer, and the storage processing which has priority over the load processing is switched to load processing priority at the time of completion of the processing.

Embodiments

A detailed description of the Embodiment of the present invention is provided hereafter with reference to the drawings.

FIG 1 is a block diagram showing an example of the present invention.

In the drawing, numeral 1 indicates a storage buffer address writing register; numerals 2, 3, and 4 indicate storage address entry registers; numerals 5, 6, and 7 indicate match detection comparators; numeral 8 indicates a main memory write pending address register; numeral 9 indicates a sweeping controller; numeral 10 indicates a storage buffer read register; numeral 11 indicates an incrementer; numeral 12 indicates a storage buffer data writing register; numeral 13 indicates an incrementer; numeral 14 indicates a store data buffer; numeral 15 indicates a main memory write pending data register; and numeral 16 indicates an incrementer.

Further, when accessing a successive load instruction that coincides with an address of an unexecuted storage instruction in relation to the main memory device while registering in the storage buffer, a construction is provided so that an unexecuted storage registered in the storage buffer can be executed in relation to the same main memory address as the access address of the load instruction which has priority over the processing of such load instruction. Furthermore, the match detection comparators 5 through 7 and the sweeping controller 9 constitutes a means that queues a load instruction until the completion of the storage processing, and the sweeping controller 9 constitutes a means that releases the

queue of the load instruction at the time of the completion of the storage processing to the same main memory address as the load instruction.

FIG 2 illustrates the details of the sweeping controller 9 in FIG 1.

Referring to FIG 2, numeral 17 indicates a flag buffer register; numeral 18 indicates an adder; numeral 19 indicates a register; and numeral 20 indicates a decrementer.

FIG 3 is a time chart of normal storage processing where (X) indicates the "Primary Example" and (Y) indicates the "Secondary Example".

Further, (a) shows a cycle; (b) is a stage IF; (c) is a stage AC; (d) is a stage AT; (e) is a stage CA; (f) is a stage EX; (g) is a stage ST; (h) is a store address entry register SA; and (i) shows a main memory.

FIG 4 is a time chart of an Embodiment of the present invention, and (a) shows a cycle; (b) is a stage IF; (c) is a stage AC; (d) is a stage AT; (e) is a stage CA; (f) is a stage EX; (g) is a stage ST; (h) is a store address entry register SA; (i) is a main memory; (j) is a write address pointer; (k) is a write data pointer; and (l) shows a read pointer.

FIG 5 is a diagram showing an example of pipeline processing.

The following describes the operation of the Embodiment shown in FIG 1 with reference to FIG 2 through FIG 5.

Referring to FIG 5 which shows an example of pipeline processing, the stages are divided into six stages from the stage IF through the stage ST. The stage IF is the process for fetching a instruction; the stage AC is the process for creating an operand; the stage AT is the process for converting a logical address to an actual address; the stage CA is the process for reading an operand; the stage EX is the process for calculating a read operand; and the stage ST is the process for storing calculation result.

An Embodiment of the present invention is described hereafter based on the pipeline stage shown in FIG 5.

First, a description in regards to the normal storage process will be given with reference to the time chart shown in FIG 3. When the storage A instruction process begins at the time of t_1 as the "Primary Example" shown in (X), the storage address for storage A at the time of t_4 is registered in the storage address entry registers 2 through 4 of FIG 1. Then, calculation is performed at the time of t_5, and data that should be stored is registered in the storage data buffer 14 shown in FIG 1 of stage ST at the time of t_6. In this way, the timing for registering an address into the storage buffer is different from the timing for registering storage data into the storage buffer regarding the storage instruction. Further, if there is a load B instruction in relation to the address B in stage AT at the time of t_6, because the information processing device with the present invention is controlled so that the load instruction process can have processing priority over the storage instruction process, the load B instruction accesses the main memory at the time of t_7, and the storage A instruction queues for processing at the storage buffer, and then, access with the main memory of the storage A instruction is performed at the time of t_8.

Next, the process of the storage C instruction begins at the time of t_9 as the "Secondary Example" shown in (Y), and when the process of the load C instruction begins at the time of t_{10}, a successive load C instruction is executed before the preceding storage C instruction is completed at the time of t_{12}.. Here, if the successive load C instruction is processed first, the wrong data may be fetched. Therefore, the preceding storage C instruction must take priority thereby allowing access the main memory of the storage C instruction at the time of t_{13} and accessing the main memory of the load C instruction at the time of t_{14}.

The present invention is a control method for the "Secondary Example" shown in (Y) of FIG 3, and a detailed description is given hereafter with reference to the Embodiment shown in FIG 1 and the time chart shown in FIG 4.

Referring to the time chart of FIG 4, when considering that each instruction is received in stage IF at the times of t_1 through t_7 in the order of storage D, storage E, storage E, storage F, load G, load H, and Load E, a storage instruction proceeds to
IF → AC → AT → CA → EX → ST, and
a load instruction proceeds to IF → AC → AT.
Regarding a storage instruction, an address of each storage instruction is registered in the stage SA ($t_4 \sim t_7$) with a timing of AT → CA. Storage data of a storage instruction is registered in the storage data buffer 14 of FIG 1 ($t_{[illeg.]} \sim t_{[illeg.]}$) with a timing of EX → ST. Then, a storage instruction where a storage address and storage data are gathered waits the timing for sweeping from the storage buffer → to the main memory. The timing of this sweeping is "the time when a load instruction does not use the main memory". Accordingly, sweeping from the storage buffer to the main memory is possible every time except the timing when a load instruction is swept from the stage AT → to the main memory.

The storage D is swept to the main memory at the time of t_7, and the main memory access of the load G is performed at the time of t_8, and the main memory access of the load H is performed at the time of T_9.

There is a load E instruction in relation to the address E in the stage AT at the time of t_9. Also, there is the storage E instruction in relation to the address E within the storage buffer. At that time, as described in the "Secondary Example", the storage instruction process takes priority over the process of the load E instruction until the unexecuted storage E instruction is completed within the storage buffer. Accordingly, the storage E is swept to the main memory at the time of t_{10}, and the subsequent storage E is swept to the main memory at the time of t_{11}. At this point, since there are no unexecuted storage E instructions within the storage buffer, the sweep of the storage buffer is inhibited in the sweeping controller 9 shown in FIG 1, and the main memory access of the load E instruction is performed at the time of t_{12}, and the main memory sweep of storage F is performed at the time of t_{13}.

Further, writing a storage instruction to the storage address entry registers 2 through 4 that are shown in FIG 1 is performed by the storage buffer address writing register 1, and writing to the storage data buffer 14 is performed by the storage buffer data writing register 12 respectively, and sweeping from the storage buffer to the main memory is performed by the storage buffer read register10.

The sweeping controller 9 understands how many unexecuted "storage instructions that must take priority over a load instruction" exist within the storage buffer. A detailed description of this sweeping controller 9 is shown in FIG 2. Referring to FIG 1, the match detection comparators 5 through 7 examine whether or not a value of each of the storage address entry registers 2 through 4 matches with a successive address value. When it matches and the successive instruction is a load instruction, a flag shall be set in the flag buffer register 17 of FIG 2. Further, the number of Totn 1 of the match detection comparators 5 through 7, in other words, the number of storage instructions that must be swept, is shown in the register 19. Then, the storage buffer is swept by the storage buffer reading register 10 of FIG 1, and when the output from the flag buffer register 17 of FIG 2 is "1", the calculation of the register 19 is performed by the decrementer 20 which is one counter, and when the register 19 becomes "0", sweeping from the storage buffer is inhibited assuming there is no storage instruction that needs to be swept within the storage buffer, and then, the control change is performed so that a load instruction can be initiated.

FIG 6 is a block diagram showing another Embodiment of the present invention.

Referring to FIG 6, the same codes are equivalent to FIG 1, and numeral 21 is a sweeping controller.

Further, when accessing a successive load instruction that matches with an address of an unexecuted storage instruction in relation to the main memory device while registering in the storage buffer, a construction is provided so that only unexecuted storage instructions registered in the storage buffer can be executed in relation to the same main memory

address as the access address of the load instruction which has priority over the processing of such load instruction.

Furthermore, the match detection comparators 5 through 7 and the sweeping controller 21 constitutes a means that queues a load instruction until the completion of the storage processing, and the sweeping controller 21 constitutes a means that releases the queue of the load instruction at the time of the completion of the storage processing to the same main memory address as the load instruction, and also a means that controls the writing so that a storage instruction, which has been executed within the storage buffer and has been written in the main memory device, is not written again in the main memory device.

FIG 7 is a drawing showing the details of the sweeping controller 21 in FIG 6.

Referring to FIG 7, numeral 22 is a decoder; numeral 23 is a validity storage instruction flag buffer; numeral 24 is an encoder; numeral 25 is a priority storage instruction flag buffer; numeral 26 is an encoder; numeral 27 is a storage buffer read assignment register; numeral 28 is a selector; numeral 29 is a storage buffer read register; numeral 30 is an adder; numeral 31 is a register; numeral 32 is a decrementer; numerals 33_1, 33_2, ... 33_n are AND circuits; and numeral 34 is a priority storage instruction signal line.

FIG 8 is a time chart of another Embodiment of the present invention, (a) shows a cycle; (b) is a stage IF; (c) is a stage AC; (d) is a stage AT; (e) is a stage CA; (f) is a stage EX; (g) is a stage ST; (h) is a store address entry register SA; (i) is a main memory; (j) is a write address pointer; (k) is a write data pointer; and (l) shows a read pointer.

The operation of the Embodiment shown in FIG 6 is described hereafter, with reference to FIG 7 and FIG 8. In addition, there is no difference in comparison to the Embodiment shown in FIG 1 regarding the pipeline processing, therefore, the description will be omitted.

First, the present invention is a control method for the "Secondary Example" shown in (Y) of FIG 3.

Referring to the time chart of FIG 8, when considering that each instruction is received in stage IF at the times of t_1 through t_8 in the order of storage D, storage E, storage F, storage E, load G, load H, load E, and storage G, a storage instruction proceeds the processing to

IF → AC → AT → CA → EX → ST, and

a load instruction proceeds to

IF → AC → AT.

Further, regarding a storage instruction, an address of each storage instruction is registered in the storage address entry registers 2 through 4 ($t_4 \sim t_7$, t_{11}) of FIG 6 with a timing of AT → CA. Furthermore, storage data of a storage instruction is registered in the storage data buffer 14 of FIG 6 ($t_6 \sim t_9$, and t_{13}) with a timing of EX → ST.

Then, a storage instruction where a storage address and storage data are gathered waits for the timing for sweeping from the storage buffer to the main memory. The timing of this sweeping is "the time when a load instruction does not use the main memory". Accordingly, sweeping from the storage buffer to the main memory is possible every time except for the timing when a load instruction is accessed from stage AT to the main memory.

The storage D is swept to the main memory at the time of t_7, and the main memory access of the load G is performed at the time of t_8, and the main memory access of the load H is performed at the time of T_9. There is a load E instruction in relation to the main memory address E in the stage AT at the time of t_9. At that time, there is a storage E instruction in relation to the main memory address E within the storage buffer. At that time, as described in the "Secondary Example" shown (Y) of FIG 3, the storage E instruction process takes priority over the process of the load E instruction until the unexecuted storage E instruction is completed in relation to the main memory from the storage buffer therein.

At that time, because there are two storage instructions in relation to the main memory address E within the storage buffer, referring to Embodiment shown FIG 6, these two storage instructions E are given priority in execution. Then, the first storage E instruction is swept to the main memory from the storage buffer at the time of t_{10}, and the second storage E instruction is swept to the main memory at the time of t_{11}. At this point, since there are no unexecuted storage E instructions within the storage buffer, the sweeping of the storage buffer is inhibited in the sweeping controller 21; the main memory access of the load E instruction is performed at the time of t_{12}; unexecuted storage F instruction is performed in relation to the main memory within the storage buffer at the time of t_{13}; and the storage G instruction is performed in relation to the main memory address G at the time of t_{14}.

Further, writing a storage instruction to the storage address entry registers 2 through 4 is performed by the storage buffer address writing register 1, and writing to the storage data buffer 14 is performed by the storage buffer data writing register 12 respectively, and sweeping to the main memory from the storage buffer is performed by the storage buffer read register 29.

The sweeping controller 21 understands how many unexecuted "storage instructions that must take priority over a load instruction" exist within the storage buffer, and which is indicated by raising a flag whether or not an instruction must be swept in relation to each word. The sweeping controller 21 controls so that only flagged storage instructions can be swept. Also, after completion of the storage processing, it is a necessary to return the value of the storage buffer read register to the word position of a storage instruction that is unexecuted in relation to the main memory within the storage buffer, and the sweeping controller 21 controls sweeping so that storage instructions that have already been swept in relation to the main memory cannot be swept again.

A detailed description of the sweeping controller 21 is shown in FIG 7.

Normal storage processing refers to the validity storage instruction flag buffer 23. This validity storage instruction flag buffer is a flag buffer to raise flag "1" on an assigned word at the time of the storage address writing and to perform reset at the time of storage buffer sweeping, and which shows whether or not a storage instruction for each word is executed in relation to the main memory.

Further, the value for the number of words of the storage buffer is set from O through N respectively in the storage buffer read assignment register buffer 27. An output signal from the validity storage instruction flag buffer 23 is encoded by the encoder 24 so that the word position of unexecuted storage instructions within the storage buffer can be assigned, and the storage buffer read register 29 is set by assigning the word position from where sweeping should be done from the storage buffer in the output signal from the storage buffer read assignment register buffer 27 through the selector 28.

When a storage instruction that has not been executed in relation to the same memory address and a load instruction are conflicting as shown in the "Secondary Example" of (Y) in FIG 3 as described above, a flag is raised for a word corresponding to the priority storage instruction flag buffer 25 under the condition where the value of match detection comparators 5 through 7 that are shown in FIG 6 in the AND circuits 33_1 through 33_n are "1" (priority storage address = successive load address); at the time of a load request processing demand; and an unexecuted storage request in relation to the main memory (output of the validity storage instruction flag buffer 23). The output signal of the priority storage instruction flag buffer 25 is decoded by the encoder 26, and this encoder 26 judges which storage instruction processing is performed, and selects the output signal from the storage buffer read assignment register buffer 27 passing through the selector 28, and sets the storage buffer read register 29. The adder 30 calculates how many storage instructions there are within the storage buffer that should be given priority to process, and sets this in the register 31. Further, every time a priority storage processing is executed, the register 31 is renewed by the decrementer 32. It is a normal storage processing when the register 31 is "0", and when register 31 is "1", an unexecuted storage instruction within the storage buffer and a load instruction are conflicted

concerning accessing to the same memory address, and it is used for a selection signal of the selector 28.

Resetting of the validity storage instruction flag buffer 23 is performed in relation to a flag of a word position that is going to be set in the storage buffer reading register 29. Further, a word position flag that has been read from the priority storage instruction flag buffer 25 is reset with the same timing. In other words, only a word position flag of an unexecuted storage instruction in relation to the main memory is "1" within the significant storage instruction flag buffer 23 as well as the priority storage instruction flag buffer 25.

Efficacy of the Invention

The present invention described above has an advantageous effect when there is a successive load instruction in relation to the storage instruction within an unexecuted storage buffer that precedes to the same main memory address, it has the ability to eliminate delayed accessing to the main memory of a load instruction through unnecessary storage processing, by switching the storage processing, which has priority over the load processing, to the load processing priority at the time of completion of all the storage processing in relation to the same memory address within the storage buffer.

The present invention also has an advantageous effect when there is a successive load instruction in relation to a storage instruction within an unexecuted storage buffer that precedes the same main memory address, and it has the ability to eliminate delayed accessing to the main memory of a load instruction through unnecessary storage processing by executing only the storage processing in relation to the same main memory address within the storage buffer, and switching the storage processing, which has priority over the load processing, to load processing priority at the time of completion of the processing.

4. Brief Description of the Drawings

FIG 1 is a block diagram showing one Embodiment of the present invention.

FIG 2 is a Drawing showing details of a sweeping controller in FIG 1.

FIG 3 is a time chart of the normal storage processing.

FIG 4 is a time chart of the Embodiment of the present invention.

FIG 5 is a diagram showing an example of the pipeline processing.

FIG 6 is a block diagram showing another Embodiment of the present invention.

FIG 7 is a Drawing showing details of a sweeping controller in FIG 6.

FIG 8 is a time chart of another Embodiment of the present invention.

1	Storage buffer address writing register
2 ~ 4	Storage address entry register
5 ~ 7	Match detection comparator
8	Main memory write pending address register
9	Sweeping controller
10	Storage buffer read register
11	Incrementer
12	Decrementer
13.....	Incrementer
14	Storage data buffer
15	Main memory write pending data register
16	Incrementer
17	Flag buffer register
18	Adder
19	Register
20	Decrementer
21	Sweeping controller
22	Decoder
23	Validity storage instruction flag buffer
24	Encoder
25	Priority storage instruction flag buffer
26	Encoder
27	Storage buffer reading assignment register
28	Selector
29	Storage buffer reading register
30	Adder
31	Register
32	Decrementer
$33_1 \sim 33_n$	AND circuit

Patent Applicant: Nihon Electric Corporation

Agent: Masaki Yamakawa (two others)

FIG 1

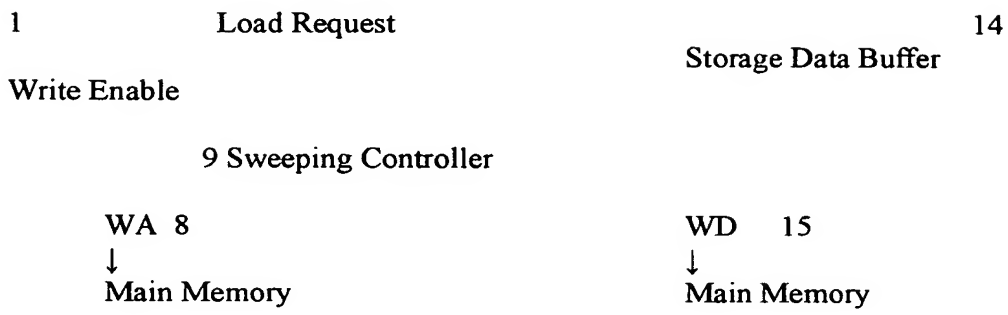
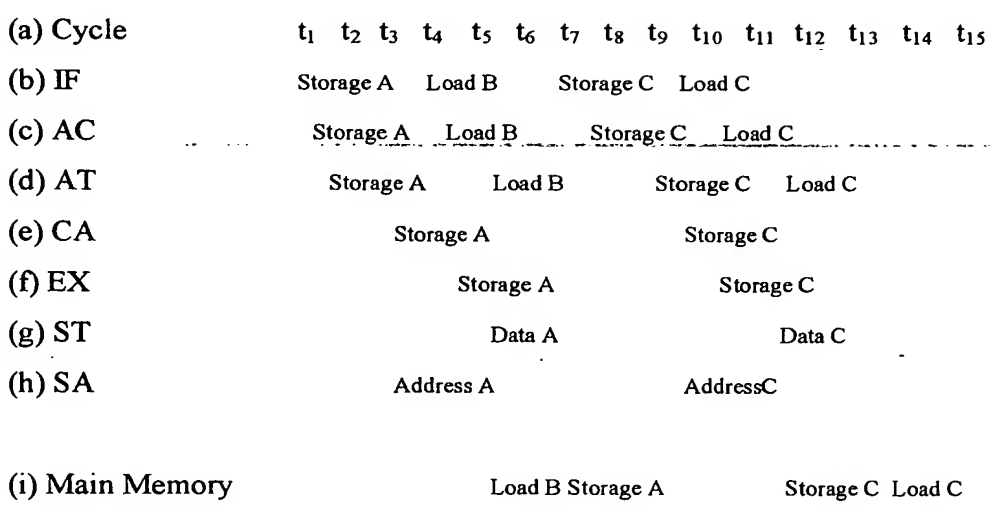


FIG 2

FIG 3



(X) (Y)

FIG 4

(a) Cycle

.

.

.

(i) Main Memory

(j) Write Address Pointer

(k) Write Data Pointer

(l) Read pointer

FIG 5

AIC

Instruction Cache

IR

Address Adder

LAR

TLB

PAR

Operand Cache

EXR

Operation Unit

RDR

Storage Buffer

WD

Main Memory

FIG 6

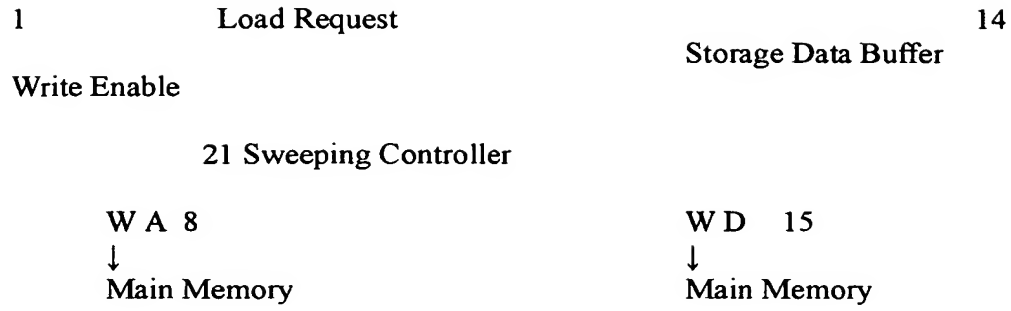


FIG 7

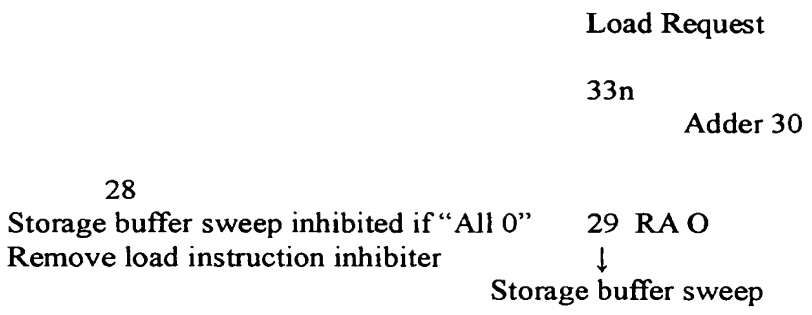


FIG 8

(a) Cycle

.

.

.

(i) Main Memory

(j) Write Address Pointer

(k) Write Data Pointer

(l) Read Pointer